Photo2ClipArt: Image Abstraction and Vectorization Using Layered Linear Gradients

JEAN-DOMINIQUE FAVREAU, FLORENT LAFARGE, ADRIEN BOUSSEAU, Inria - Université Côte d'Azur

(a) Input image and segmentation (b) Multi-layer abstraction and vectorization (c) Editing

Fig. 1. Given a segmented bitmap image as input (a), our method generates an abstract, layered vector clipart, where each layer is filled with an opaque or semi-transparent linear color gradient (b). By expressing the image as a stack of linear color gradients, our vector graphics reproduce the visual style of traditional cliparts and are easy to edit (c). In this example, we turned the lady bug blue (top) and replaced its dots by little stars (bottom). The black dots in the segmentation indicate opaque regions selected by the user to initialize or constrain our algorithm.

We present a method to create vector cliparts from photographs. Our approach aims at reproducing two key properties of cliparts: they should be easily editable, and they should represent image content in a clean, simplified way. We observe that vector artists satisfy both of these properties by modeling cliparts with linear color gradients, which have a small number of parameters and approximate well smooth color variations. In addition, skilled artists produce intricate yet editable artworks by stacking multiple gradients using opaque and semi-transparent layers. Motivated by these observations, our goal is to decompose a bitmap photograph into a stack of layers, each layer containing a vector path filled with a linear color gradient. We cast this problem as an optimization that jointly assigns each pixel to one or more layer and finds the gradient parameters of each layer that best reproduce the input. Since a trivial solution would consist in assigning each pixel to a different, opaque layer, we complement our objective with a simplicity term that favors decompositions made of few, semi-transparent layers. However, this formulation results in a complex combinatorial problem combining discrete unknowns (the pixel assignments) and continuous unknowns (the layer parameters). We propose a Monte Carlo Tree Search algorithm that efficiently explores this solution space by leveraging layering cues at image junctions. We demonstrate the effectiveness of our method by reverse-engineering existing cliparts and by creating original cliparts from studio photographs.

CCS Concepts: • **Computing methodologies** → *Non-photorealistic rendering*;

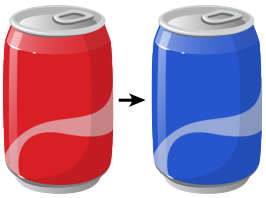
Additional Key Words and Phrases: image abstraction, image stylization, vector graphics, vectorization, layers, transparency, color gradient

2017. 0730-0301/2017/11-ART180 $15.00
https://doi.org/10.1145/3130800.3130888

ACM Reference format:
Jean-Dominique Favreau, Florent Lafarge, Adrien Bousseau. 2017. Photo2ClipArt: Image Abstraction and Vectorization Using Layered Linear Gradients. *ACM Trans. Graph.* 36, 6, Article 180 (November 2017), 11 pages.
https://doi.org/10.1145/3130800.3130888

1 INTRODUCTION

Vector cliparts are widely used in graphic design to represent images in a compact and editable manner. Cliparts are also greatly appreciated for the abstract, stylized look produced by their constituent parametric paths and color fills. Our goal in this paper is to help novice users create such stylized vector artworks from photographs.



Our approach is inspired by the observation that skilled artists produce rich yet abstract cliparts by stacking, or *layering*, few parametric vector paths. For example, the inset shows the clipart of a red soda can where the texture, shadows and highlights are all represented as semi-transparent layers filled with constant colors or color gradients[1]. Changing the color of the bottom layer is sufficient to obtain a convincing blue can. Following a similar strategy, our approach seeks to decompose the input image into a small number of semi-transparent layers, each layer filled with a constant color or a two-color linear gradient. By favoring a simple interpretation of the input, our method jointly abstracts and vectorizes the image to produce a stylized, editable clipart.

We formulate our approach as a joint labeling and fitting problem, where we want to assign each pixel to one or more layers such

[1]We rasterized all vector arts shown in the paper to ensure that transparency is well displayed in different pdf viewers. Please refer to supplemental materials for the original vector files.

that approximating the resulting layers with linear color gradients yields a good reconstruction of the input image. We balance this reconstruction error with a simplicity term that penalizes the use of too many layers. However, the resulting optimization involves both discrete unknowns (the pixel assignments) and continuous ones (the color gradient parameters). Finding the optimal solution of such a problem would require evaluating all possible pixel assignments, which is impractical. Our main technical contribution is a stochastic algorithm that explores this solution space efficiently.

Our algorithm builds on several key ideas to drastically decrease the number of configurations it needs to evaluate. Our first idea is to exploit image junctions, which provide strong local layering and transparency cues [Sayim and Cavanagh 2011]. In particular, we assume that a X-junction occurs when a semi-transparent layer runs across two other layers, which gives us a small number of possible configurations. Our second idea is to use the local decompositions obtained from X-junctions to initialize a region-growing approach, which progressively assigns new pixels to their best-fitting existing layers and only create a new layer when no good assignment is found. If no X-junction exists, we ask the user to indicate an opaque region in the image to initialize the region-growing process. However, growing the layers in a greedy manner would quickly lead us to poor local minima of our energy formulation. Our third idea is to explore multiple assignments concurrently by building a tree of candidate configurations. Each node of the tree corresponds to an intermediate decomposition where only a subset of the image pixels have been explored, while the leafs of the tree correspond to all the possible decompositions of the entire image. We present a Monte Carlo Tree Search algorithm [Browne et al. 2012] guided by image junctions to quickly identify the low-energy branches of this tree. We further accelerate the search by pre-segmenting the image into smooth color regions, turning our approach into a region labeling rather than pixel labeling problem.

While our algorithm can produce plausible vector cliparts automatically, we allow users to refine the result by indicating opaque regions and regions that should contain the layers of another region. We demonstrate our method by reverse-engineering bitmap cliparts and by creating a range of new cliparts from photographs.

2 RELATED WORK

Image vectorization. Most vectorization algorithms represent color images with a single layer. Commercial tools such as Adobe Illustrator's Image Trace [Adobe 2013] only support constant color fills and as such require users to balance over-segmentation with quantization artifacts. Lecot and Lévy [2006] were among the first to attempt vectorizing images with parametric gradients (linear and quadratic). Follow-up work introduced more complex gradient representations, such as gradient meshes [Sun et al. 2007], diffusion curves [Orzan et al. 2008], subdivision surfaces [Liao et al. 2012]. While these methods are able of vectorize highly realistic images, the complexity of the resulting single-layered representations hinders subsequent editing. Richardt et al. [2014] tackled this challenge by proposing an interactive method to convert bitmaps into opaque and semi-transparent linear vector layers. However, their method requires extensive user intervention to iteratively select the semi-transparent regions in a front-to-back order. In contrast, our algorithm automatically identifies and orders opaque and transparent regions by favoring a simple interpretation of the image. Similarly to Richardt et al., we focus on linear gradients as they are simpler to extract and edit than gradient meshes and diffusion curves, while reproducing the distinctive abstract look of vector cliparts.

Our objective of favoring a simple vectorization is similar in spirit to the work by Favreau et al. [2016] who convert line drawings into a small yet accurate set of Bezier curves, and to the work by Jeschke et al. [2011] and Zhao et al. [2017] who optimize for diffusion curves with simple color and shape respectively. Our originality is to target layered color images, which greatly increases the dimensionality of the problem and requires us to design a custom exploration mechanism of the solution space.

Image abstraction. Researchers in non-photorealistic rendering have proposed a variety of image filters to abstract and stylize photographs. Popular methods include the use of scale-space filtering [DeCarlo and Santella 2002] and edge-aware filtering [Winnemöller et al. 2006] to remove low-contrast details. However, such image abstraction filters are usually applied independently of image vectorization. In contrast, we obtain abstract, stylized cliparts by expressing the image with a small number of parametric color gradients. A similar idea of restricting the image formation model to achieve abstraction has been explored by Gerstner et al. [2012], who produce pixel art by converting a photograph into a low-resolution image with a reduced color palette.

Image decomposition. While our goal is to produce vector graphics, our work is also related to layered decomposition of bitmaps. In particular, Tan and colleagues [2015; 2016] decompose artworks into semi-transparent layers to perform a range of editing operations. Their first method takes as input a time lapse video of the artwork being painted [Tan et al. 2015], and as such is not applicable in our context. In a follow-up work [Tan et al. 2016], they observe that the dominant colors of an image correspond to the vertices of the convex hull of the image in RGB space. This analysis allows them to decompose the image into a stack of smooth semi-transparent layers, each layer corresponding to one of the dominant colors with a per-pixel transparency. Similarly, Aksoy et al. [2017] propose a soft segmentation algorithm that decomposes an image into layers with per-pixel transparency, although they associate each layer with an RGB normal distribution rather than a constant color. We adopt a very different approach by representing our layers with linear color gradients amenable to vectorization. Our more constrained representation allows us to express each layer with a handful of parameters and a binary mask rather than with one continuous unknown per pixel. Our energy formulation also allows us to explicitly minimize the complexity of the output and to automatically find the layer ordering.

Our problem is also related to other ill-posed image decompositions such as image matting [Smith and Blinn 1996], reflection separation [Levin et al. 2004] and intrinsic images [Bell et al. 2014]. However, a major difference between these methods and our work is that they aim at separating only two rather than multiple layers, being foreground and background or reflectance and shading.

Nevertheless, several such algorithms make the decomposition well-posed by penalizing complexity via a prior on sparse image gradients [Levin et al. 2004] and few refectances [Bell et al. 2014]. Similarly, we favor a small number of layers, each represented by a constant color or a linear gradient.

Depth ordering and transparency from junctions. Perceptual studies emphasize the role of image junctions in the perception of occlusion and transparency [Metelli 1974; Sayim and Cavanagh 2011]. In particular, T-junctions provide strong cues of local ordering between opaque layers [Jia et al. 2012; Liu et al. 2013] while luminance and chrominance patterns at X-junctions have been used for extracting transparent layers from images [D'Zmura et al. 1997; Singh and Huang 2003]. We follow a similar approach to identify transparent and opaque layers at X-junctions. However, while prior work derives heuristic rules to reason about layer ordering and transmittance, we use an energy formulation to solve for the layer configuration that best reconstructs the input image. Our energy also favors simple interpretations, which allows us to deal with configurations where no X-junctions occur, such as when a transparent region is entirely surrounded by another region.

3 PROBLEM FORMULATION

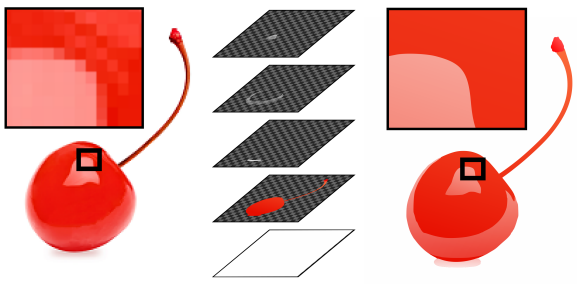
Our goal is to estimate a multilayer vectorial representation of an input image I, where each layer is composed of

- A supporting domain D covering a subset of pixels from the input image,
- A color gradient function $C(p)$ that associates an RGB color to each pixel $p \in D$,
- An opacity gradient function $A(p)$ that associates an opacity to each pixel $p \in D$. We set $A(p) = 0$ when $p \notin D_n$.

We synthesize the output image I_n from a n-layer representation by recursively α-blending the ordered layers

$$I_n(p) = A_n(p)C_n(p) + (1 - A_n(p))I_{n-1}(p) \tag{1}$$

where C_n and A_n are the color gradient function and opacity gradient function of layer n. Figure 2 illustrates this representation.



(a) Input (b) Layers (c) Output

Fig. 2. Multilayer representation. Our goal is to represent the input image (a) as a stack of opaque and semi-transparent layers (b) such that α-blending the layers using Eq. 1 reproduces well the input (c).

Linear gradients. While the above formulation is general, in this work we restrict the color and opacity gradients to linear forms with the same orientation, which corresponds to the 2-stop linear gradient of the SVG standard. The color and opacity gradients functions are expressed as $C(p) = c_0 + c_1 O^t p$ and $A(p) = a_0 + a_1 O^t p$ where c_0 and c_1 (resp. a_0 and a_1) are color vectors (resp. opacity scalars) and O is the orientation vector. While linear gradients are less expressive than more complex primitives, such as gradient meshes [Sun et al. 2007] and diffusion curves [Orzan et al. 2008], they are easier to manipulate thanks to their small number of parameters and are supported by most vector graphics software. In addition, many vector artists employ linear gradients as a means to simplify and stylize the image. Restricting our algorithm to linear gradients allows us to reproduce this characteristic style of vector cliparts.

Pre-segmentation. Recovering the color, opacity and supporting domain of each layer at each pixel is a formidable task. We reduce the complexity of this task by pre-segmenting the image into smooth regions and by imposing that all pixels of a region share the same layers. While our algorithm produces convincing results from automatic segmentation, we achieved our best results using user-assisted segmentation as detailed in Section 5. In addition, we apply a 3-pixel wide erosion on each region to exclude border pixels since those often contain a mixture of colors from neighboring regions. We process these pixels in a separate step once the multi-layer representation is computed, as detailed in Section 4.6.

3.1 Energy formulation

We denote by $\mathbf{x} = (D_1, C_1, A_1, ..., D_n, C_n, A_n)$, an output vectorial representation composed of n layers. We define three criteria to measure the quality of a configuration $\mathbf{x}$:

- Faithfulness to input data - the reconstructed image I_n has to be similar to the input image I,
- Simplicity of the decomposition - the number of layers should be minimized to yield a compact and editable representation,
- Simplicity of opacity functions - semi-transparent layers should be favored over opaque ones to reduce occlusion.

Based on these criteria, we formulate an energy U of the form

$$U(\mathbf{x}) = (1 - \lambda)U_{fidelity}(\mathbf{x}) + \lambda U_{simplicity}(\mathbf{x}) \tag{2}$$

where $U_{fidelity}$ measures the faithfulness to input data and $U_{simplicity}$ accounts for the overall simplicity of the reconstruction. The parameter $\lambda \in [0, 1]$ weights the two terms.

We define $U_{fidelity}$ as the RGB error under L_2 norm between the input image I and the reconstructed image I_n:

$$U_{fidelity}(\mathbf{x}) = \frac{1}{|I|} \sum_{p \in I} \|I(p) - I_n(p)\|_2^2. \tag{3}$$

We express $U_{simplicity}$ as

$$U_{simplicity}(\mathbf{x}) = \frac{1}{N} \sum_{l=1}^{n} w_l \tag{4}$$

where N is the maximal number of layers and w_l penalizes opacity functions according to their simplicity. We set $w_l = 1$ for semi-transparent layers and $w_l = \beta > 1$ for opaque layers. Figure 2

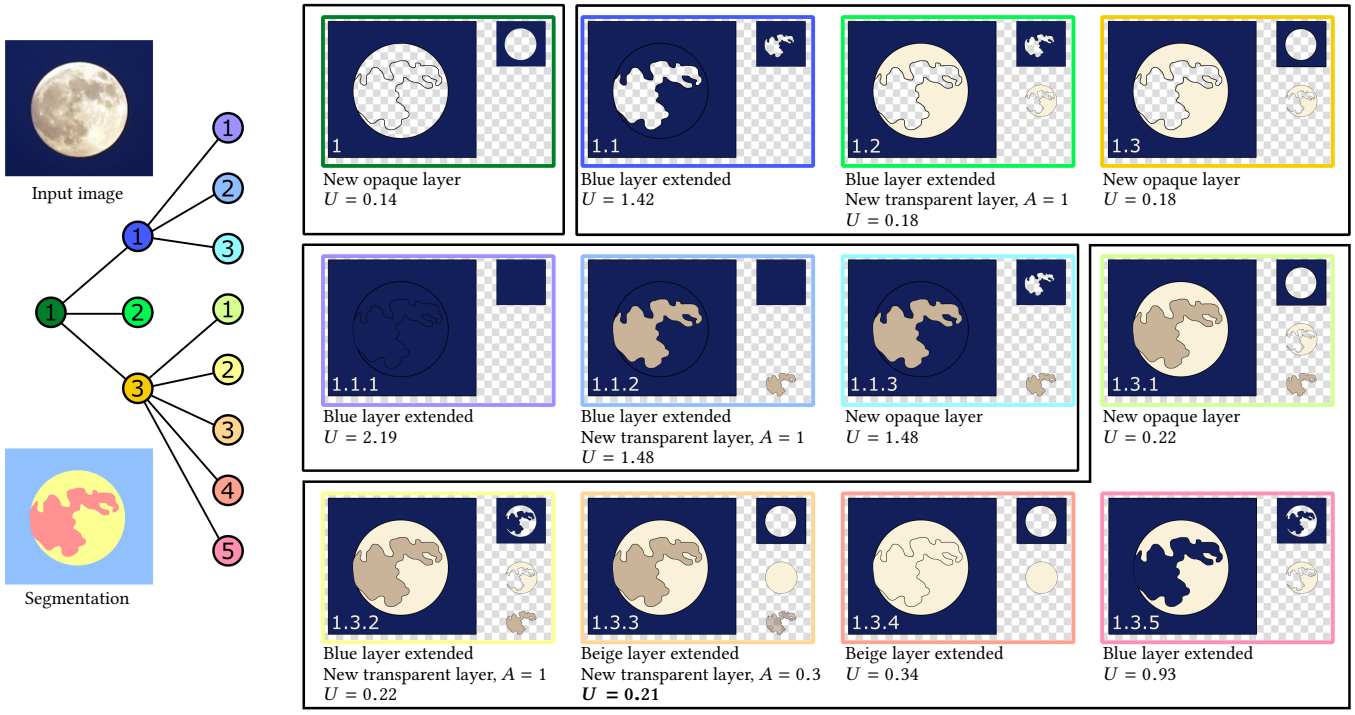


Fig. 3. Overview of our exploration mechanism on a simple example. Given an input image and its region segmentation, we cast the exploration of the solution space as a tree search. Each node of the tree corresponds to an intermediate solution where only a subset of the regions has been decomposed. Each branch of the tree expands an intermediate solution by adding a region to the decomposition. The new region can be assigned to one or several existing layers, as well as to a new layer. Leafs of the tree correspond to complete decompositions. We use Monte Carlo Tree Search to quickly reach low-energy leaves. In this example, the best solution is reached at leaf 1.3.3, where the moon is represented as an opaque beige layer with a semi-transparent brown layer on top. Note that in some configurations, such as node 1.2, a region can be assigned to a new semi-transparent layer which receives an opacity value of 1 after optimization. We convert such layers to opaque when this situation occurs.

illustrates the intuition behind this strategy: the highlight on the cherry could be either interpreted as an opaque pink region surrounded by a red region, or as a semi-transparent white region over a red region. Setting $\beta > 1$ favors the latter interpretation, which facilitates subsequent editing such as changing the color of the cherry. We used $\beta = 1.2$ in all our examples. In addition, we convert a semi-transparent layer to opaque if its estimated opacity is above 0.999.

4 EXPLORATION MECHANISM

Searching for the configuration that minimizes energy U is a complex optimization problem which combines discrete variables (the number of layers and their supporting domains) and continuous variables (the parameters of the linear gradient functions). While the number of layers can be infinite, we can reasonably assume that there is at most as many layers as input regions. Yet, since a region can appear in at most all N layers, finding the global minimum would require estimating the continuous variables for 2^{N^2} layer assignments, which is impractical despite this upper bound.

We address this combinatorial challenge by devising a scalable exploration strategy of the solution space, which we illustrate on a simple example in Figure 3. Our exploration starts from initial decompositions of a few regions, which we obtain by exploiting transparency assumptions on X-junctions, or by asking the user to indicate one or more opaque regions. We then expand these decompositions to the regions adjacent to the ones already decomposed. Each new region can either be assigned to one or more existing layers, or to a new layer, for which the color gradient is estimated. We thus obtain a set of possible assignments of the new regions, each forming a decomposition that can be further expanded to adjacent regions. We can represent the ensemble of decompositions generated by this approach as a tree, where each node corresponds to an intermediate decomposition, which branches to multiple decompositions after an expansion step[2]. The leafs of this tree correspond to all the 2^{N^2} possible decompositions, and our goal is to find a low-energy leaf without exploring the entire tree.

We propose a stochastic algorithm for fast exploration of the solution space, which can be seen as a form of Monte Carlo Tree Search (MCTS) [Browne et al. 2012]. The main idea of this method is to build the solution tree incrementally and asymmetrically using a balance between exploration of the space and exploitation of the already explored configurations. Figure 4 illustrates the four successive operations performed at each iteration of an MCTS method:

[2]In theory, the tree of solutions is actually a directed acyclic graph because multiple paths can yield the same configuration. However, in practice such cases are very rare because our acceleration heuristics trim many branches of that graph.

(i) selection of a node according to a tree policy, (ii) expansion of the tree by adding child nodes, (iii) evaluation of the quality of the added child nodes (also called *reward*), and (iv) update of the tree policy by back-propagation to the root. Algorithm 1 summarizes this process.

Our main technical contribution resides in defining efficient expansion and reward operations that exploit characteristics of our problem. We denote by $\mathbf{x}_{/t}$ an intermediate decomposition restricted to the regions visited between the root and node t. We measure the energy $U(\mathbf{x}_{/t})$ by restricting I to the visited regions (Eq. 3), and by setting the maximal number of layers N to the number of visited regions (Eq. 4).

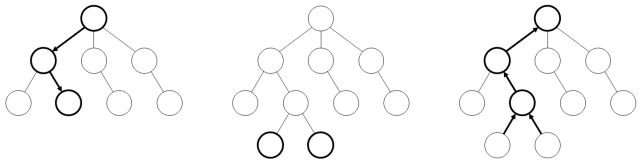


Fig. 4. Exploration mechanism by Monte Carlo Tree Search (after [Browne et al. 2012]). A node is first selected using a tree policy (section 4.1). New nodes are then added (section 4.2) and their corresponding energy are computed (section 4.3). The energy of the new nodes is back-propagated to their parent node to discourage exploring bad configurations (section 4.4).

Algorithm 1 Exploration mechanism

Initialize child nodes of the root (sec 4.5)
repeat
  Node selection: select a leaf node using tree policy (sec 4.1)
  Expansion: generate child nodes (sec 4.2)
  Reward: compute energy of child nodes (sec 4.3)
  Back-propagation: update tree policy (sec 4.4)
until stopping criterion (sec 4.5)

4.1 Node selection

The tree policy seeks to favor the exploration of high-quality configurations. To do so, each child node t_i of a node t is associated to a parent-to-child probability that is proportional to its energy and the energy of its siblings

$$P_{t\to t_i} = \frac{\exp(-U(\mathbf{x}_{/t_i}))\mathbb{1}_{\{t_i\in t^\star\}}}{\sum_{t_j\in t^\star}\exp(-U(\mathbf{x}_{/t_j}))} \tag{5}$$

where $t^\star$ is the set of child nodes of t and $\mathbb{1}$ is the indicator function. The algorithm selects a terminal node for expansion according to the product of parent-to-child probabilities between the root and the node, i.e. the overall probability of reaching this configuration.

We further accelerate the search by restricting $t^\star$ to the k children nodes with the lowest energies. The parameter k offers a balance between spreed and accuracy of the algorithm. We set k to 2 in all our experiments.

4.2 Tree expansion

Once a candidate terminal node is selected, the tree expansion step generates its child nodes by adding one or more regions to the ones already visited. We first present a naive generator that simply adds one region with all its possible layer assignments. We then detail a more efficient generator that exploits a transparency assumption on X-junctions to add multiple regions at once. We detect X-junctions as cliques of order 4 in the adjacency graph of the regions.

Single-region expansion. This generator inserts an unvisited region to the decomposition associated with the selected node. This region is chosen randomly among the regions adjacent to the ones already present in the decomposition. We assume that the new region can be part of any layer of its adjacent regions in the decomposition, as well as part of a new layer. Note that we do not consider layers of non-adjacent regions, which drastically reduces the number of child nodes while ensuring that each layer contains a single connected component by construction. Given the M layers from the adjacent regions, a naive enumeration gives 2^{M+1} child nodes. However, since opaque layers occlude all layers below them, we can discard all but one of the configurations that have the same visible layers, which further reduces the number of child nodes.

X-junction expansion. This generator only applies on the nodes for which the decomposition contains part of an X-junction, and expands the decomposition to all four regions forming the X-junction. We build on the assumption that an X-junction results from the boundary of a semi-transparent layer crossing the boundary of two other layers [Sayim and Cavanagh 2011]. This assumption yields a limited set of possible interpretations, which depends on how many of the four regions are already present in the decomposition of the selected nodes:

- None of the regions of the junction have been decomposed. This configuration occurs when we initialize the algorithm, i.e. when the X-junction is used to expand the root of the tree. In the absence of other information, we make the additional assumption that the junction is formed by one semi-transparent layer over two opaque layers, which yields 4 possible configurations illustrated in Figure 5.
- One or more regions of the junction are already in the decomposition. We can deduce that the boundary between an unknown region and a known one is either caused by the end of a known layer or the beginning of a new one. Denoting M the number of layers in an adjacent known region, we obtain $M+1$ configurations if the unknown region is in the 4-neighborhood of the known one (i.e. the two regions are separated by one boundary), and $M+2$ configurations if the unknown region is in the 8-neighborhood of the known one (i.e. the two regions are separated by two boundaries).

We call the X-junction generator in priority since it allows faster expansion of the tree than the single-region generator.

4.3 Reward

Given the layer assignment of a child node, we can estimate its energy along with the color and opacity gradient function of each layer by minimizing Equation 3. However, this optimization is highly

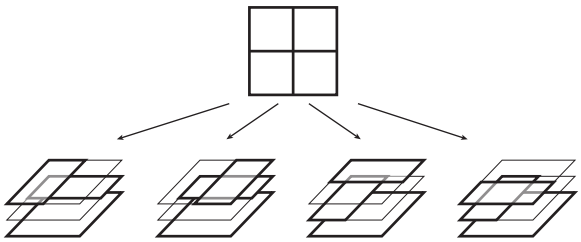


Fig. 5. We assume that X-junctions are formed by a semi-transparent layer crossing two other layers. When none of the layers are known, we assume that the two bottom layers are opaque, yielding 4 possible configurations.

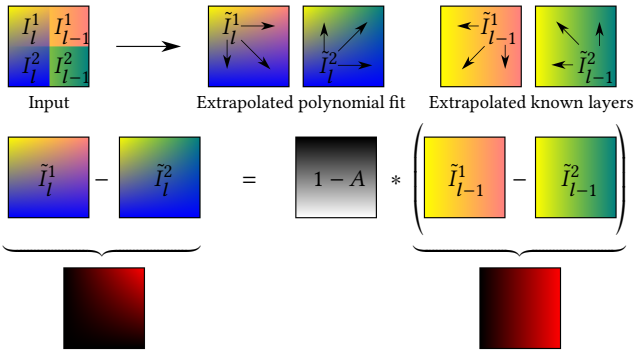


Fig. 6. Derivation of the opacity of a new layer at an X-junction. Let us assume that the new layer is shared by the two regions I_l^1 and I_l^2. Furthermore, region I_l^1 shares all of its other layers with region I_{l-1}^1, while region I_l^2 shares all of its other layers with region I_{l-1}^2. We exploit this redundancy to deduce the opacity of layer l. We first extend each region over the other regions to know their values at each pixel (top). Regions I_{l-1}^1 and I_{l-1}^2 are trivially extended since their layers are known and have a linear form. We extend regions I_l^1 and I_l^2 by fitting a polynomial on their pixel colors. We can then compute a per-pixel opacity A by simple arithmetic (bottom).

non-linear, which requires the use of an iterative algorithm like Newton-Raphson with a good initialization of the solution. We propose three strategies to avoid computing such a costly optimization for most visited nodes of the tree.

Layer re-use. Our first strategy to speed up the evaluation of Equation 3 is to re-use the color and opacity gradients of a parent node when present in a child node. In other words, we extend the known gradient functions over the new regions without re-fitting them. While this approach is approximate, it yields a significant gain in performance with a marginal loss in visual quality, as evaluated in Section 5.

Exploitation of X-junctions. As detailed above, expanding the decomposition at an X-junction only yields configurations with at most one new semi-transparent layer. In addition, our assumption on the nature of X-junctions tells us that this semi-transparent layer covers the boundary between two regions with known layers. Our strategy is to deduce from this setup a configuration where the semi-transparent layer is observed over two known colors, which makes its estimation well-posed [Richardt et al. 2014; Smith and Blinn 1996].

Denoting l the unknown layer, each of the two regions on which it appears gives us equations of the form $I_l(p) = A_l(p)C_l(p) + (1 - A_l(p))I_{l-1}(p)$ where I_l is the input image from which we have substracted all semi-transparent layers above l and I_{l-1} is the image formed by all layers below l, which are known for the two regions considered. The challenge is now to express these equations over the same pixels, so that we can remove the term $A_l(p)C_l(p)$ by substitution and leave $A_l(p)$ as the only unknown.

Figure 6 illustrates our algorithm on a toy example, where the new layer covers two regions I_l^1 and I_l^2, which share layers with their known neighboring regions I_{l-1}^1 and I_{l-1}^2 respectively. We first extend each region over all three other regions so that all regions share the same spatial domain. Regions that are already in the decomposition are easy to extend since all their layers are known and have a parametric form. However, regions covered by the new layer are still in a bitmap form. Our solution consists in approximating each such region with a polynomial function using a least square fit on all pixels. Since each layer is represented by a linear gradient, a new region can at best be represented by a polynomial of degree $M+1$, where M is the number of layers shared with the adjacent regions. We then extend the resulting parametric functions over the other regions to know their values over all pixels. We now have the necessary ingredients to form two equations at each pixel

$$\tilde{I}_l^1(p) = A_l(p)C_l(p) + \big(1 - A_l(p)\big)\tilde{I}_{l-1}^1(p) \quad (6)$$

$$\tilde{I}_l^2(p) = A_l(p)C_l(p) + \big(1 - A_l(p)\big)\tilde{I}_{l-1}^2(p), \quad (7)$$

where $\tilde{I}_j^i$ denotes the extended version of region I_j^i. Subtracting the two equations gives us an expression where $A_l(p)$ is the only unknown

$$\tilde{I}_l^1(p) - \tilde{I}_l^2(p) = \big(1 - A_l(p)\big)\big(\tilde{I}_{l-1}^1(p) - \tilde{I}_{l-1}^2(p)\big). \quad (8)$$

Once $A_l(p)$ is known the minimization of Equation 3 becomes quadratic in $C_l(p)$ and has a unique solution.

Note that since each region of the image is visited multiple times during the tree search, we only compute its polynomial representation at a given degree the first time it is needed, and keep it in memory for later use.

Sub-sampling. Our last strategy to reduce computational burden is to use a sparse uniform random sampling of the input image pixels when fitting the color and opacity gradients of the layers. However, using very few samples raises the risk of making the method very sensitive to image noise. Fortunately, we can leverage the polynomial approximation of the image introduced above as a means to remove high frequency content that cannot be captured by the layers. While each polynomial needs to be computed from all pixels of its region, this is a one time computation which is quickly amortized over all the nodes where the layers of a given region need to be estimated.

To sum-up, we first generate for each node a number of configurations, which form its child nodes. When a child node does not contain any new layer, computing its energy U by Eq. 2 is trivial

Fig. 7. Layered cliparts produced with our method from three studio photographs and a bitmap diagram. We used manual segmentation to properly delineate small and blurry regions (letters on the stop button, highlights on the tomato) and to ignore spurious details (arrows on the chart).

since the parameters of all layers are known. When new layers are involved, the parameters of their color and opacity gradient functions must be estimated first. If the new layer is opaque and below known semi-transparent layers, or if the new layer is part of an X-junction, we can estimate the gradient functions as a quadratic minimization problem that has a unique solution. Otherwise, we run the Newton-Raphson algorithm. Both the quadratic minimization and the Newton-Raphson algorithm are computed with a sparse sampling of the input image.

To further reduce the complexity of the tree, we remove the child nodes whose energies are one order of magnitude higher than the best child node.

4.4 Back-Propagation

Once the energy of new child nodes is computed, we update the tree policy to favor the visit of low energy configurations. We back-propagate information on child node energy towards the root by giving to each node t the lowest energy of its child nodes

$$U(\mathbf{x}_{/t}) \leftarrow \min_{t_i \in t^\star} U(\mathbf{x}_{/t_i}). \tag{9}$$

4.5 Initialization and stopping criterion

We initialize the exploration by generating one root node for each X-junction, using as initial decompositions of these junctions the configurations that minimize $U_{fidelity}$. In the cases where no X-junction exists, we ask the user to seed the search by selecting an opaque region in the image. We display a black dot over the selected region of the segmentation for each result where such indication was provided. We stop the exploration once we have reached $4N$ leaves of the tree, with N the number of regions in the image, and keep the decomposition with the lowest energy as our final solution. We adjusted this stopping threshold experimentally by running our algorithm multiple times on the same image using a high threshold, keeping track of the number of leaves visited before reaching the best configuration.

4.6 Finalization

As a reminder, we excluded pixels on the boundary of each region of the segmentation, since those pixel are often corrupted by blur or anti-aliasing. The last step of our algorithm assigns each such pixel to one of its neighboring regions based on goodness of fit. In addition, when an opaque layer is surrounded by another opaque layer, we position the surrounding layer below the surrounded one and extend it to cover the hole, which is occluded by the surrounded layer. As an example, the red layer of the lady bug in Figure 1 extends below the black dots.

Finally, we vectorize the supporting domain of each layer using Potrace [Selinger 2017].

4.7 User interaction

We offer users several means of controlling our method. First, we expose the global λ parameter, which balances fidelity to the input with simplicity of the output. We found that clean bitmap cliparts can be decomposed using a low λ value of 0.1, while photographs often require a higher value of 0.5 to abstract away small details, non-linear color variations, and image noise. We also allow users to increase β to favor the extraction of semi-transparent layers, although we used the default value for all results in this paper.

Second, we allow users to impose that a region of the image be expressed as a single opaque layer. We implement this constraint during the tree search by only generating child nodes with opaque layers over the selected region. We display a black dot over the selected regions of the segmentation for each result where such additional constraint was provided.

Finally, users can also control the appearance of the decomposition via the input segmentation, for instance to approximate fine details or complex color variations with a unique region.

5 EXPERIMENTS

Figure 1 and Figure 7 show results of our method on a variety of illustrations and studio photographs. Our approach is especially effective at expressing highlights and shadows with semi-transparent layers, which facilitates subsequent manipulation like color and shape editing, as shown in Figure 1(c).

We now compare our method to existing work in the field and evaluate the impact of parameters and performance.

5.1 Comparisons with existing work

Figure 9 compares our method with the work by Tan et al. [2016] and Richardt et al. [2014]. The method by Tan and colleagues targets the different application of bitmap color editing. As a result, while their method succeeds in separating the image into layers based on the dominant colors of the image, each layer is a bitmap with a constant color and spatially-varying transparency rather than a vector path filled with a parametric gradient.

Our goal and approach is more similar to the work by Richardt et al., and our method produces similar results to theirs. The main difference between the two methods resides in the user workflow. The interactive workflow of Richardt et al. requires users to extract each layer of the decomposition one by one, in a front to back order. Since their layer extraction takes between a few seconds to a minute per layer, their results took between several minutes to an hour to create, as detailed in Section 5 of their paper. In contrast, users of our system simply have to provide a segmentation of the image, which takes a few minutes with an interactive tool, and optional indications of opaque regions before letting our algorithm produce the entire decomposition within seconds. Another difference resides in the family of gradients supported by the two methods. Our method extracts 2-stop gradients, while Richard et al. also support 3-stop gradients, but requires users to indicate the number of stops for each region. This is why their method extracts each color band of the cone as a single 3-stop gradient while we extract them as two 2-stop gradients (Figure 9, 3rd row). In addition, Richard et al. represent opaque layers with gradient meshes, while we only use linear gradients. Their gradient mesh better captures contrast on the opaque layer of the cylindrical mug (Figure 9, 4th row).

5.2 Impact of parameters

The main parameter of our algorithm is λ, which controls the amount of abstraction of our vectorization. Figure 8 details its impact on the vectorization of a bitmap clipart. A low value of λ reproduces the input as closely as possible with linear gradients. Increasing λ favors the use of semi-transparent layers at the price of more approximate rendition of the original colors. A high value of λ abstracts away details in an effort to reduce the number of layers.

5.3 Impact of pre-segmentation

Figure 10 compares the output of our method with different input segmentations generated with a Mean Shift algorithm [Comaniciu and Meer 2002] and a manual segmentation. Our method produces visually similar vectorizations with two different automatic segmentations. In particular, multiple regions are merged to form a few layers thanks to our simplicity term. However, we obtain more stylized cliparts using manual segmentations.

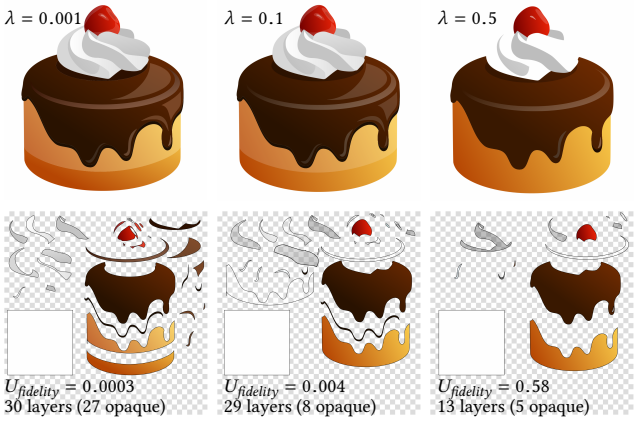


Fig. 8. Impact of parameter λ. A low value of λ favors accurate reproduction of the input, but produces a decomposition dominated by opaque layers (left). Increasing λ turns many of the opaque layers semi-transparent (middle). A high value reduces the number of layers, at the price of missing details (right).

Image	Exploration	Finalization	Total	# regions	# clicks
Lady bug	28s	6s	35s	26	3
Cherry	5s	2s	7s	8	1
Stop	16s	12s	28s	23	2
Tomato	9s	6s	16s	19	1
Diagram	12s	1s	13s	11	3
Battery	44s	15s	60s	24	2
Cake	71s	148s	220s	30	1
Egg	7s	2s	9s	6	1
Red shoes	5s	11s	17s	11	1
Purair	16s	4s	21s	20	1
Cone	44s	3s	48s	27	0
Cup	17s	16s	33s	11	1
Wine	13s	13s	26s	11	0
House	337s	159s	496s	105	3

Table 1. Our method generates layered vectorizations within seconds. The cake image is one of the slowest to process because it has many regions and no X-junctions.

5.4 Performance

Table 1 provides timings for several of our results, measured on a computer equipped with a 3rd generation core i7 processor and 16Gb DDR3 memory. Our algorithm took less than a minute to produce each decomposition, which allows an interactive workflow where users can refine the result by adding a new opacity constraint or modifying the segmentation if necessary. The last column of Table 1 detail the number of opaque regions selected by the user to initialize the optimization.

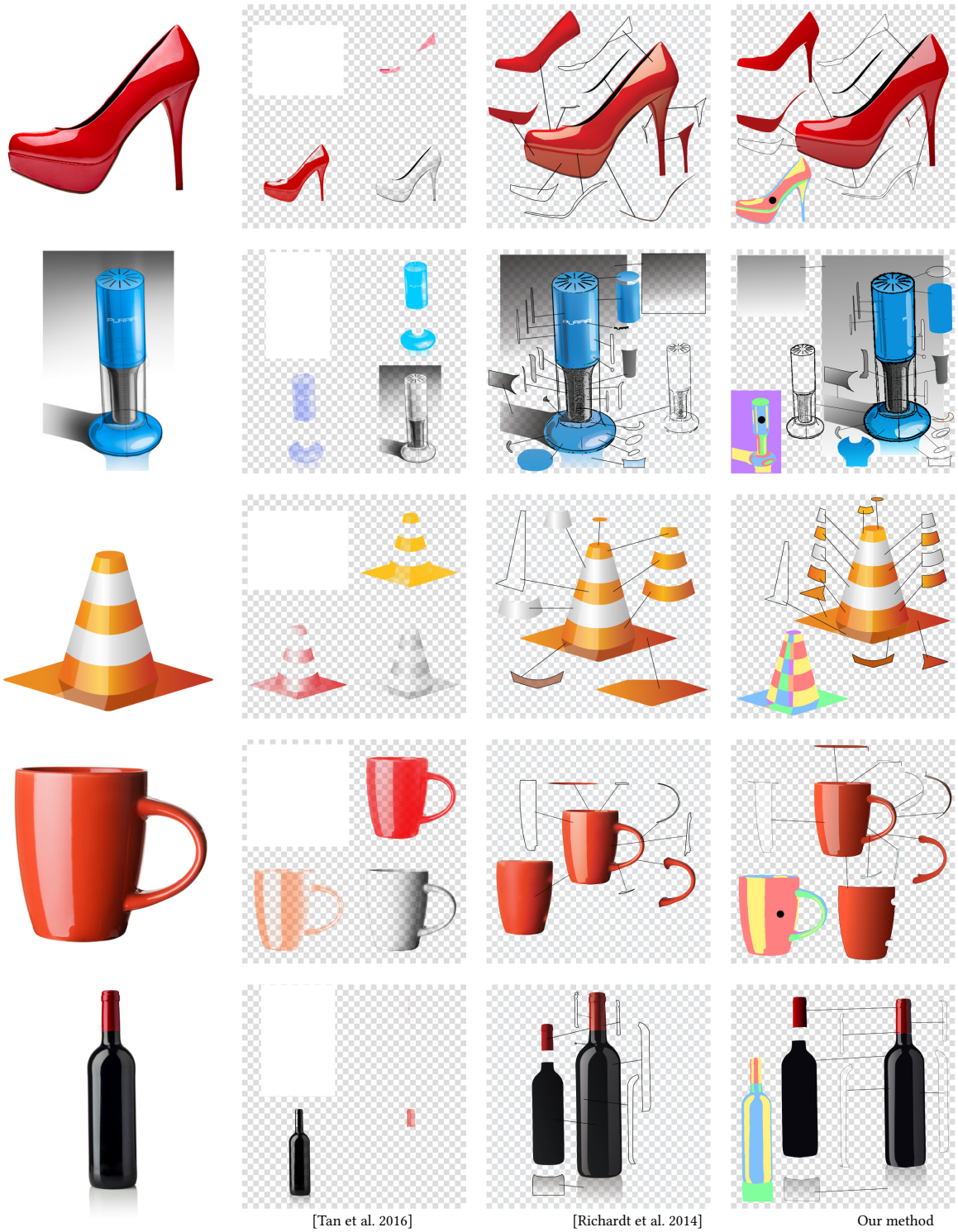


Fig. 9. Comparison with existing decomposition methods. The layers generated by [Tan et al. 2016] are bitmaps filled with a constant color and spatially varying transparency, which is suitable for color editing but not for other applications of vector graphics. Our results are similar to the ones by [Richardt et al. 2014], although our method requires less user intervention and solves for the layer parameters more efficiently. All our results were produced with a manual segmentation, except the cone (3rd row).

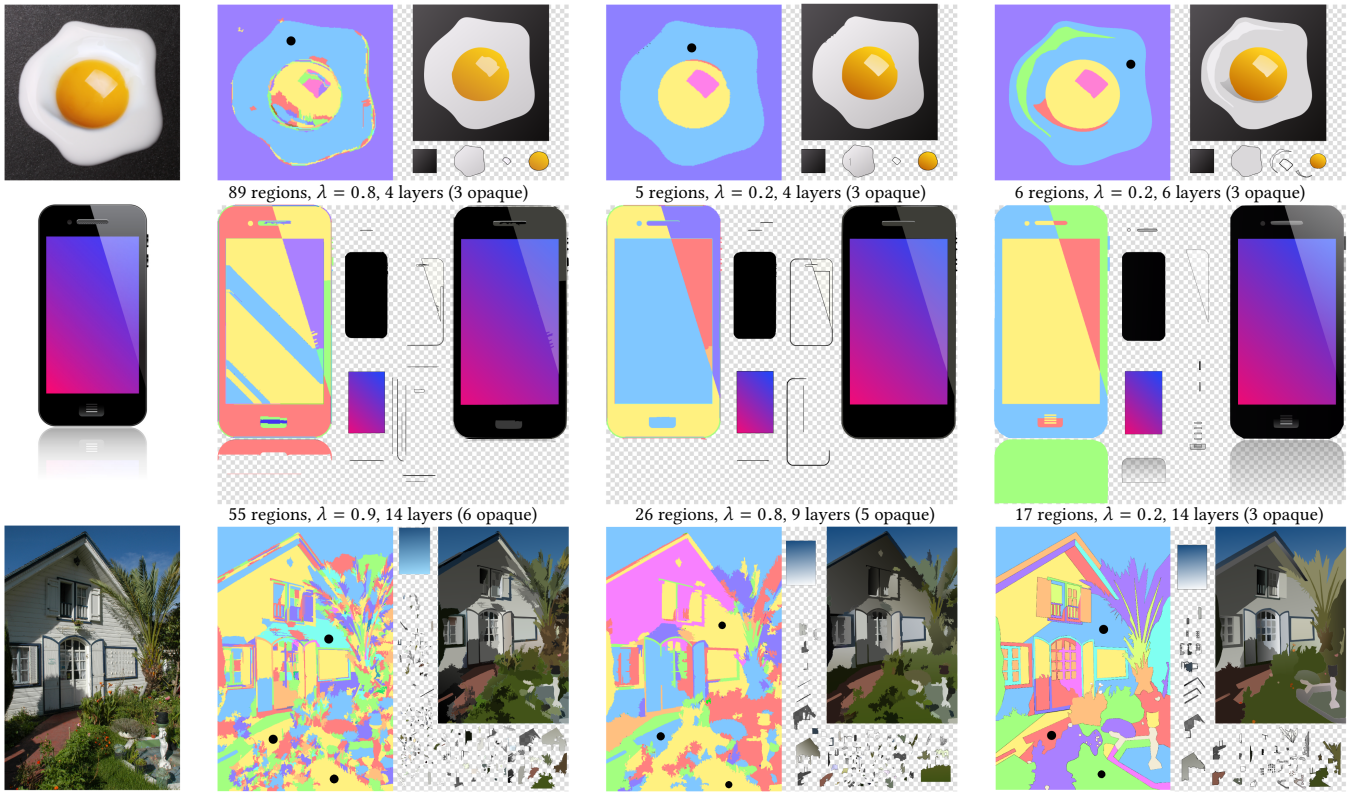


Fig. 10. While our method produces consistent results with different automatic segmentations (b,c), we achieved our best results by refining the segmentation by hand (d). In particular, while our method can fuse small regions to reduce the number of layers, spurious high-contrast details remain, such as thin highlights along the boundary of the smartphone. In contrast, a manual segmentation allows users to remove unwanted details while preserving others, such as an extra highlight on the egg, the button of the smartphone, and the window tiles on the house. Note that the shadows on the house and ground are extracted as semi-transparent layers.

Table 2 compares several versions of our algorithm where we removed important features. In particular, exploiting X-junctions greatly speeds up the exploration of the solution space, while increasing the maximum number of child nodes does not yield a significant increase in quality. We also implemented re-fitting of all layers for each child node, which increases timing by two orders of magnitude without much impact on the energy.

X-junctions	yes	yes	no	no
k	2	4	2	2
Layer re-use	yes	yes	yes	no
Energy U	1.00	1.04	0.870	0.869
Time (s)	14	28	307	29890
Number of visited nodes	318	332	54358	52357

Table 2. Ablation study. We evaluated the performance of our algorithm on the smartphone image (Figure 10) after removing some of its key components. Removing the exploitation of X-junctions increases computation time by one order of magnitude, while re-fitting all layers for each child node further increases computation by two orders of magnitude without significant change in the energy. Similarly, augmenting the maximum number of child nodes k slows down the algorithm without much impact on the energy. We provide the resulting vector cliparts as supplemental material, they are all visually similar.

5.5 Limitations

Our choice of restricting the image representation to linear gradients is key to achieve joint abstraction and vectorization. However, our current implementation does not support variants of linear gradients, such as radial gradients, although such primitives could be included in the configurations evaluated by the tree search. More advanced primitives like gradient meshes would be more difficult to integrate since they cannot be trivially expanded to adjacent regions, which is a key assumption made by our algorithm.

Our method targets piecewise-smooth images and as such reaches its limits in the presence of texture. A low-contrast texture is averaged-out if segmented as a single region, while high-contrast textures can be segmented in multiple small regions, which result in a cluttered clipart with many layers. The house in Figure 10 illustrates how our approach performs on a complex natural image with textures.

We speed up our algorithm by assuming that X-junctions are caused by transparency. This assumption breaks on texture patterns such as a checker-board, where many X-junctions are not

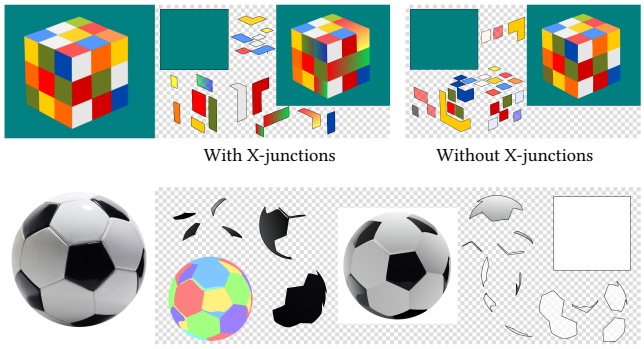


Fig. 11. Limitations. Top: We accelerate our algorithm by assuming that each X-junction is produced by a semi-transparent layer crossing two other layers. This assumption is not valid on the Rubik's cube, where there are multiple X-junctions between tiles of different colors. Exploiting our assumption on this example forces the method to include transparent layers that do not reproduce well the input. Bottom: Decomposing an image into opaque and transparent layers is especially ambiguous when the layers share the same color. In this example, the algorithm cannot make the distinction between a dark shadow over a white panel and a white highlight over a dark panel. As a result, some black panels are interpreted as white and vice-versa.

due to transparency. Figure 11(top) illustrates the behavior of our algorithm in such cases, with and without the exploitation of X-junctions. While our method produces a faithful vectorization when X-junctions are not exploited, exploiting them forces the algorithm to include transparent layers, which reduces the accuracy of the reconstruction.

Inverting Equation 1 is an ill-posed problem, especially when the foreground and background layers contain the same colors. This ambiguity is illustrated in Figure 11 where the soccer ball is only composed of shades of gray. As a result, while the decomposition found by our optimization captures well the appearance of the input image, it does not properly separate the white and black panels of the ball because it cannot make the distinction between a dark shadow over a white panel and white highlight over a dark panel.

6 CONCLUSION

While image vectorization has received significant attention in the computer graphics community, very little work has been done on reproducing the style and layer structure of traditional vector cliparts. Motivated by the ubiquity of linear gradients in vector art, we have presented an algorithm based on Monte Carlo Tree Search to jointly decompose an image into layers and approximate these layers with linear gradients.

Our method takes a segmented image as input and does not attempt to modify the shape of the segmented regions, apart from fusing small segments to crate bigger ones. However, the regions produced by automatic segmentation algorithms often have more intricate shapes than the ones created by vector artists. An interesting direction for future research would be to jointly simplify shape and color during the vectorization process, potentially by including a shape simplicity term in our energy formulation.

ACKNOWLEDGMENTS

This work was supported in part by the ERC starting grant D^3 (ERC-2016-STG 714221) and by research and software donations from Adobe. Many thanks to Tan and colleagues [2016] for making their code available. Image credits: ladybug by Alex Staroseltsev, soda can by Altagracia Art, cherry by M. Unal Ozmen, stop button by Photo Melon, chart by Allies Interactive, tomato by bajinda, battery by Photo Melon, cup by George Dolgikh, shoe by Picsfive, wine bottle by Dmitri Gristsenko, egg by Valentina Razumova, house by Stefano Ember, smart phone by Gor Grigoryan, soccer ball by Le Do, all on Shutterstock.com. Air purifier by Spencer Nugent on sketch-a-day.com, cake by vectorsme on openclipart.org, moon by stux on pixabay.com.

REFERENCES

Adobe. 2013. Adobe Illustrator Image Trace, http://blogs.adobe.com/adobeillustrator/2013/07/image-trace-in-illustrator-a-tutorial-and-guide.html. (2013).
Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Transactions on Graphics* 36, 2 (2017).
Sean Bell, Kavita Bala, and Noah Snavely. 2014. Intrinsic Images in the Wild. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014).
C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012).
Dorin Comaniciu and Peter Meer. 2002. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 24, 5 (2002).
Doug DeCarlo and Anthony Santella. 2002. Stylization and Abstraction of Photographs. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21, 3 (2002).
Michael D'Zmura, Philippe Colantoni, Kenneth Knoblauch, and Bernard Laget. 1997. Color transparency. *Perception* 26, 4 (1997).
Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: a Global Approach to Line Drawing Vectorization. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)* (2016).
Timothy Gerstner, Doug DeCarlo, Marc Alexa, Adam Finkelstein, Yotam Gingold, and Andrew Nealen. 2012. Pixelated Image Abstraction. In *Proc. International Symposium on Non-photorealistic Animation and Rendering*.
Stefan Jeschke, David Cline, and Peter Wonka. 2011. Estimating Color and Texture Parameters for Vector Graphics. *Computer Graphics Forum (Proc. Eurographics)* 30, 2 (April 2011), 523–532.
Zhaoyin Jia, A. Gallagher, Yao-Jen Chang, and Tsuhan Chen. 2012. A learning-based framework for depth ordering. In *IEEE Computer Vision and Pattern Recognition*.
Gregory Lecot and Bruno Lévy. 2006. Ardeco: automatic region detection and conversion. In *Proceedings of the Eurographics Symposium on Rendering Techniques*.
Anat Levin, Assaf Zomet, and Yair Weiss. 2004. Separating reflections from a single image using local features. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition*.
Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. 2012. A Subdivision-Based Representation for Vector Image Editing. *IEEE Transactions on Visualization and Computer Graphics* 18, 11 (2012).
Xueting Liu, Xiangyu Mao, Xuan Yang, Linling Zhang, and Tien-Tsin Wong. 2013. Stereoscopizing Cel Animations. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 32, 6 (2013).
Fabio Metelli. 1974. The perception of transparency. *Scientific American* (1974).
Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27, 3 (2008).
Christian Richardt, Jorge Lopez-Moreno, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. 2014. Vectorising Bitmaps into Semi-Transparent Gradient Layers. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 33, 4 (2014).
Bilge Sayim and Patrick Cavanagh. 2011. The art of transparency. *i-Perception* 2 (2011).
Peter Selinger. 2017. Potrace. (2017).
Manish Singh and Xiaolei Huang. 2003. Computing layered surface representations: an algorithm for detecting and separating transparent overlays. In *IEEE Computer Vision and Pattern Recognition*.
Alvy Ray Smith and James F. Blinn. 1996. Blue screen matting. *SIGGRAPH* (1996).
Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. 2007. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26, 3 (2007).
Jianchao Tan, Marek Dvorožňák, Daniel Sýkora, and Yotam Gingold. 2015. Decomposing Time-Lapse Paintings into Layers. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (2015).
Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-space Geometry. *ACM Transactions on Graphics* 36, 1 (2016).
Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. 2006. Real-time Video Abstraction. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 25, 3 (2006).
S. Zhao, F. Durand, and C. Zheng. 2017. Inverse Diffusion Curves using Shape Optimization. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2017).